RSL-IL: An Interlingua for Formally Documenting Requirements

David de Almeida Ferreira, Alberto Rodrigues da Silva
INESC-ID, Instituto Superior Técnico (IST), Lisbon, Portugal
{david.ferreira, alberto.silva}@inesc-id.pt

***Abstract*—Despite being the most suitable language to communicate requirements, the intrinsic ambiguity of natural language often undermines requirements quality criteria, specially clearness and consistency. Several proposals have been made to increase the rigor of requirements representations through conceptual models, which encompass different perspectives to completely describe the system. However, this multi-representation strategy warrants significant human effort to produce and reuse such models, as well as to enforce their consistency. This paper presents RSL-IL, a Requirements Specification Language that tackles the requirements formalization problem by providing a minimal set of constructs. To cope with the most typical Requirements Engineering concerns, RSL-IL constructs are internally organized into viewpoints. Since these constructs are tightly integrated, RSL-IL enables the representation of requirements in a way that makes them formal enough for being tractable by a computer. Given that RSL-IL provides a stable intermediate representation that can improve the quality and enables requirements reuse, it can be regarded as a requirements interlingua. Also, RSL-IL can be used as a source language within the context of model-to-model transformations to produce specific conceptual models. To illustrate how RSL-IL can be applied in a real project, this paper provides a running example based on a case study.**

***Index Terms*—Requirements Specification Language, Information Extraction, Requirements Modeling, Requirements Verification, Requirements Transformations.**

I. Introduction

Requirements Engineering (RE) is about reaching a *shared understanding* between two different domains: the business and the technical stakeholders. This bridge can only be established through an *effective communication* between these two parties [1]. The adverse consequences of disregarding the importance of early RE activities are well-known [2]. At this early project stage, when most requirements are being developed, stakeholders must deal with highly abstract notions, and reason solely at such a conceptual level [1]. Therefore, even the smallest misunderstandings can significantly affect the delivered system in terms of its *fit-for-purpose*. In turn, the rework required to fix this misalignment with the *real* requirements of business stakeholders frequently cause the software development project to become challenged, or even fail [3]. Thus, RE must foster a *common mindset* between all the involved stakeholders to mitigate these interpretation problems regarding the business needs that must be addressed.

For achieving an effective communication, *everyone should be able to communicate by means of a common language*, and natural language provides the foundations for such a language. Natural language is regarded as universal (i.e., it is expressive enough to describe any problem), and humans are proficient at using it to communicate with each other. Thus, natural language has minimal adoption resistance as a requirements documentation technique [1]. However, although *natural language is the most common and preferred form of requirements representation* [4], it also exhibits some intrinsic characteristics that often present themselves as the root cause of many requirements defects, such as ambiguity, inconsistency, incompleteness, and incorrectness [1]. From these causes, we emphasize *ambiguity* and *inconsistency* because avoiding – or, at least, mitigating – them requires a great deal of human effort due to the large amount of information to process when combined with inadequate tool support, namely to perform the linguistic analysis of requirements. In turn, demanding such a human effort causes the requirements documentation activity to be time-consuming and error-prone. Also, although incorrectness (by definition) cannot be fixed without human validation [5], we consider that the tasks required to mitigate the negative effects of both ambiguity and inconsistency (and also incompleteness, even if only partially) can be automated if requirements are expressed in a suitable language.

In this paper we present RSL-IL, a formal Requirements Specification Language (RSL), which was designed to actively address most of these specification problems through *requirements formalization*, focusing on requirements semantics instead of only relying on their metadata (i.e., requirement attributes and traceability relations). Dealing with requirements at a semantic level with a computer-processable notation enables the automation of some requirements verification tasks, such as ambiguity resolution, consistency checking, and completeness enforcement.

The structure of this paper is as follows. Section II overviews RSLingo [6], the broader approach that relies on RSL-IL to formally represent requirements that have been originally stated in natural language by business stakeholders. Section III introduces *Online Experts* as the running example that is used to better explain the constructs provided by RSL-IL. Also, this section overviews how these constructs are organized into specific RSL-IL viewpoints, which enable the specification of requirements at two distinct abstraction levels: the Business and the System Levels. Section IV compares RSL-IL with other RSLs, and discusses its advantages, namely the enforcement of requirements consistency, ambiguity resolution, and completeness. Finally, Section V concludes this paper and presents some ideas for future work.

II. Background

Considering the premises introduced in the previous section, regarding the current practices of the requirements documentation activity, we consider that the quality of requirements specification still strongly depends on the expertise of whoever is performing this activity. Given that most RE activities are still manually performed and involve a large amount of information, to produce a high quality requirements specification one requires an experienced requirements engineer with a vast skills set. However, to avoid large discrepancies in the results of the RE process, we advocate that the *quality* of requirements specifications and the *productivity* of the requirements documentation activity can be increased through the *formalization* of requirements. The computer-processable requirements specifications that are obtained through such a formalization process enable the automation of some manual verifications – which must be performed during the requirements documentation activity – thus relieving requirements engineers from the burden of manually handling a large amount of information to identify requirements quality problems. Additionally, the achieved degree of requirements formalization can, in turn, be employed to generate complementary artifacts that better support RE activities which must rely on human cognition, such as requirements validation.

A. Requirements Formalization Problems

As discussed ahead in Section IV – where RSL-IL is compared with other RSLs –, neither the currently available RSLs emphasize a strong RE orientation (to address RE concerns without additional complexity), nor do they ensure a higher degree of formality that can be devised in an unbiased/automatic manner. For instance, more formal RSLs imply an intermediate step in which technical stakeholders are required to translate requirements into their formal notations, and then translate the created models back to natural language so that business stakeholders are able to understand their meaning. Thus, this led to the creation of RSL-IL, a formal language that provides constructs to straightforwardly represent RE-oriented concepts, such as: terms, stakeholders, goals, requirements, entities, relations, actors, functions, states, events, and uses cases. Although it can be used as a standalone RSL, RSL-IL is part of a broader approach that aims to perform linguistic analysis and formalize requirements originally stated in natural language by addressing them at a deeper semantic level.

B. The RSLingo Approach

RSLingo is an information extraction [7] approach based on linguistic patterns, and follows a multi-language strategy based on two languages: RSL-PL (Pattern Language) for defining linguistic patterns [8], and RSL-IL (Intermediate Language) that acts as a formal requirements *interlingua* [6].

RSLingo is based on lightweight Natural Language Processing (NLP) techniques that preprocess requirements textual representations. After enriching the surface requirements text with the required linguistic features, RSLingo follows an information extraction mechanism based on a pattern alignment algorithm [9] that can be used to exploit the syntactic–semantical alignment of words within a natural language sentence. This technique allows one to capture requirements-related information with the linguistic patterns defined in RSL-PL. In turn, through previously declared transformations between RSL-PL and RSL-IL, one is able to automatically generate formal and semantically equivalent RSL-IL specifications. For instance, Listing 1 illustrates one of such declarative transformations that operationalize the RSLingo approach, namely it describes how to map a RSL-PL pattern into a template-like RSL-IL snippet that is filled with the pattern's captures, when a match against natural language requirement text occurs.

Listing 1. Example of a mapping rule from RSL-PL into RSL-IL.

```
[ [?<ent> W(DET) W(N) "entity" ] "consists" "of" S(':') [?<attr> W(DET) W(N) ]
      S('.') ]
=>
{
  (DOMAIN
    (ENTITY id:!gen_id(${ent}.2) name:${ent}.2
      (ATTRIBUTE name:${attr}.2)))
}
```

If the mapping rule presented in Listing 1 is selected to be applied by the information extraction algorithm – because the pattern alignment algorithm determines that its RSL-PL pattern is the most adequate (i.e., the one with the highest similarity score) among those provided by the remaining mapping rules –, the result of applying it to a natural language requirement sentence, such as "A question entity consists of: a 'question text' (a text field).", would yield a result similar to the one presented in Listing 2.

Listing 2. Result of applying the transformation rule illustrated in Listing 1.

```
(PROJECT (...)
  (GLOSSARY (...))
  (STAKEHOLDERS (...))
  (GOALS (...))
  (SYSTEM id:"sys−adsp" name:"AIDSPortugal" (...)
    (REQUIREMENTS (...))
    (DOMAIN (...))
    (BEHAVIOR (...))
    (SYSTEM id:"sys−onln−expt" name:"Online Experts" (...)
      (REQUIREMENTS (...))
      (DOMAIN
        (ENTITY id:"ent−qstn" name:"question"
          (ATTRIBUTE name:"question text")))
      (BEHAVIOR (...)))))
```

Since the obtained RSL-IL specifications are amenable to be processed by computers, they enable the automatic verification of some requirements quality criteria, such as *clearness*, *consistency*, and *completeness*. The requirements formalization process supported by RSLingo, as well as the part played by RSL-IL, are depicted in Figure 1.

It is noteworthy that such a decoupling of linguistic concerns from requirements formalization issues – attained with this multi-language strategy – addresses the previously mentioned requirements specification problems without introducing new

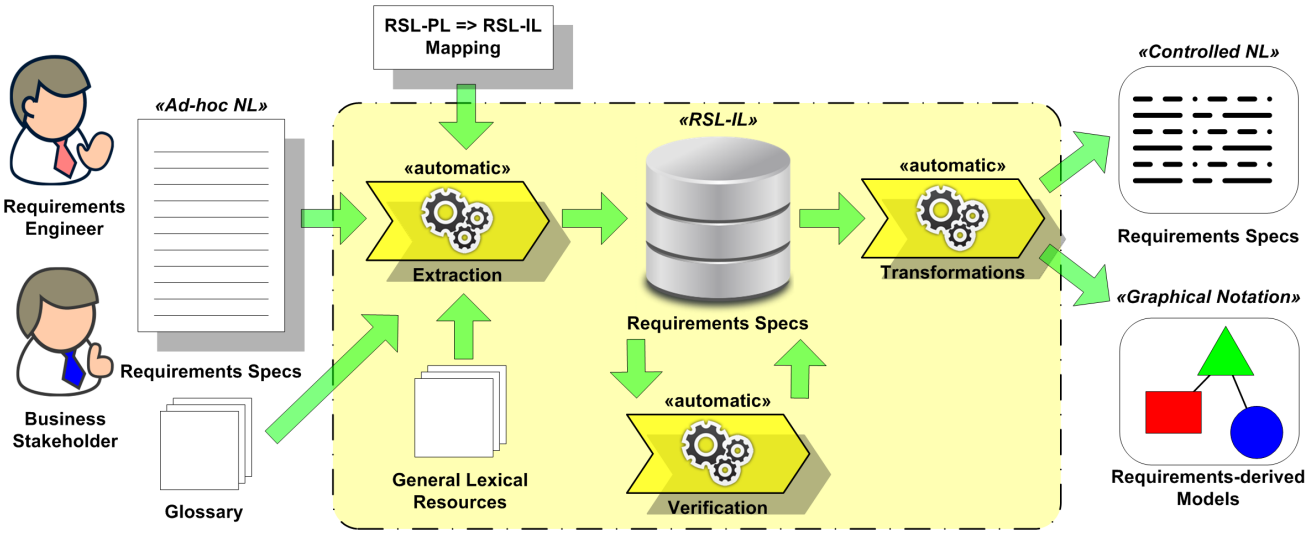

Figure 1. Overview of the RSLingo approach.

and disruptive languages that business stakeholders must learn (which, in turn, could hinder them from directly authoring and validating their own requirements).

III. The RSL-IL Language

The goal of the "formal layer" introduced by RSL-IL within the RSLingo approach is to force the *projection* of natural language requirement representations, which have a higher degree of expressiveness but are often ambiguous, into a more restricted and domain-specific representation that is tractable to reasoning and enables further computations upon requirements, at the expense of losing some detail (which can be considered negligible due to the RE orientation of RSL-IL).

A. Running Example

This example is meant to assist the reader in better understanding the RSL-IL constructs, and also to shed some light on the advantages of its application in a real-world scenario, by presenting snippets of RSL-IL concrete syntax pertaining to a requirements specification developed for a case study that was carried out to validate RSLingo. Whenever relevant, we also provide the respective excerpts of the original requirement specification documented in natural language.

This case study was conducted within the context of an industry project developed by a portuguese IT company. Part of the scope of this project was to develop an *Online Experts* feature for a larger web application[1], which is devoted to support a community of Portuguese-speaking patients infected with HIV/AIDS and/or Hepatitis viruses.

B. Viewpoints, Interrogatives, and Levels

Since RSL-IL was designed to cope with the most common RE concerns regarding the specification of software-intensive systems [1], it provides several constructs that can be logically arranged into *viewpoints* according to the specific RE concerns they address. To help RSL-IL users choose which elements to use, these viewpoints can be further organized according to the *interrogatives* that express the answers they seek to provide (the typical "five Ws and one H" questions, depicted as the columns of Figure 2) and the abstraction *levels* they are focused on (the horizontal layers of the same figure). This classification schema purposefully follows a top-down approach: it begins with higher-level and business-oriented concepts, and then progressively delves into lower-level and software-oriented concepts, thus emphasizing the two different abstraction levels addressed by RSL-IL.

The rationale underlying this organization is to ensure that RSL-IL covers all the abstraction levels related with requirements documentation, connecting both upstream with the business context, and downstream with the system concerns. Additionally, this arrangement allows one to use only some RSL-IL viewpoints in an independent manner, depending on the target audience or the task at hand.

C. The RSL-IL Business Level

In order to properly understand the business context for which the system-of-interest must provide a cost-effective and suitable solution, it is essential to document certain business-related concerns, namely: (1) the *concepts* that belong to the business jargon; (2) the *people* and *organizations* that can influence or will be affected by the system-of-interest; and (3) the *objectives* of business stakeholders regarding the value that the system-of-interest will bring. Bearing in mind these concerns, business level requirements encompasses the following RSL-IL viewpoints: (1) `Terminology`; (2) `Stakeholders`; and (3) `Goals`, respectively.

Terminology. Creating a glossary (i.e., a collection of definitions for relevant *terms*) is a highly recommended practice [1]. As depicted in Figure 3, this viewpoint provides constructs that enable the creation of project-specific glossaries, which can

[1]http://www.aidsportugal.com (accessed on January 25th, 2013)

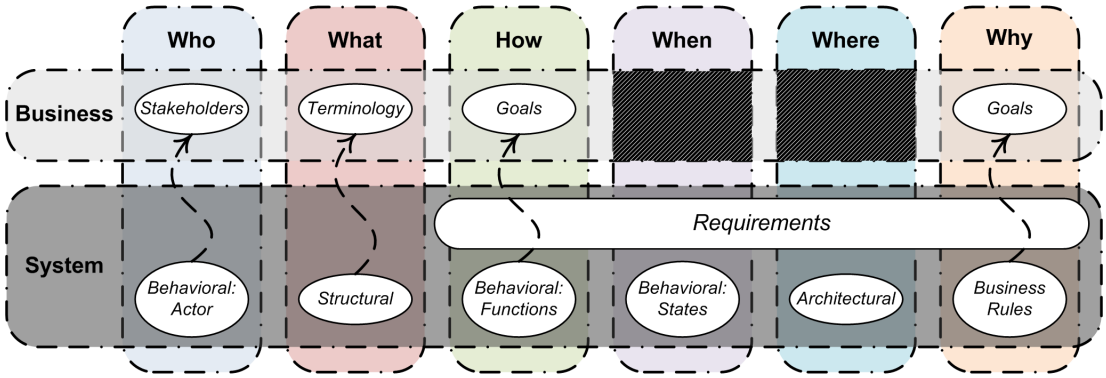

Figure 2. Classification of RSL-IL viewpoints: levels versus interrogatives.

significantly reduce the negative effects of natural language's imprecision through ambiguity resolution techniques.

The little effort required to identify and properly define the most relevant `Terms` of the *universe of discourse* of business stakeholders (specially domain experts) brings significant advantages, because these `Terms` refer to the most important real world notions to be considered. In fact, `Terms` are the *building blocks* (basic units of meaning) for any description about the "as-is", or prescription about the "to-be". Despite containing relatively few constructs, the `Terminology` viewpoint is the cornerstone for all the other RSL-IL viewpoints.

Listing 3 illustrates a partial RSL-IL specification of the `Terms` employed in the natural language requirements specification of the case study.

Listing 3. Excerpt from the Terminology viewpoint of the case study.

```
(PROJECT id:"prj−aidsportugal" name:"AIDSPortugal" description:"This
      project consists in the development, configuration, and migration of the
      AIDSPortugal web site."
  (GLOSSARY (...)
    # stakeholder terms (, a subset of)
    (TERM id:"trm−mdcl−spct" word:"medical specialist" definition:"[a
          modifier] Relating to the study or practice of medicine. [a person]
          practices one branch of medicine."
          synset:"medical.a.01;specialist.n.02") # also refers to an entity and an
          actor
    (TERM id:"trm−ptnt" word:"patient" definition:"[a person] A person who
          requires medical care." synset:"patient.n.01")
    # architectural terms (, a subset of)
    (TERM id:"trm−onln−expt" word:"Online Experts" definition:"A module of
          {AIDSPortugal} that supports the interaction between {patients} and
          {medical specialists}." pos:@noun)
    # entity terms (, a subset of)
    (TERM id:"trm−qstn" word:"question" definition:"[a communication] An
          instance of questioning." synset:"question.n.01")
    # actor terms (, a subset of)
    (TERM id:"trm−mdrr" word:"moderator" definition:"[a person] Someone
          who presides over a forum or debate." synset:"moderator.n.03")))
```

Note that the constructs of this viewpoint were designed to be tightly integrated with WordNet [10] with the purpose of enabling the reuse of the definitions and linguistic relations encoded in this lexical database. Whenever the `synset` attribute is provided, it means that the value of the `definition` attribute was extracted from WordNet.

Stakeholders. This viewpoint is devoted to the most important source of requirements: *business stakeholders*. Without backward links to these sources, one cannot ensure that the delivered software is a suitable solution because either: (1) vital information might be missing, since an important `Stakeholder` (or a group of them) was not identified early in the project; or (2) one is unable to determine the reason why a requirement was originally stated, or even to clear out its *true meaning* in subsequent phases. In short, the constructs provided by the `Stakeholders` viewpoint (illustrated in Figure 3) are crucial to explicitly represent the organizational background of `Stakeholders` (as exemplified in lines 3–7 and 11–17 of Listing 4) to better reason about their "stake" on the system-of-interest and, in case a conflict occurs, who has the authority to decide the solution to be followed.

Listing 4. Excerpt from the Stakeholders viewpoint of the case study.

```
(PROJECT (...)
  (STAKEHOLDERS
    # SIDAnet organization members
    (ORGANIZATION id:"stk−sdnt" role:"client organization" name:"SIDAnet"
          category:@business.customer
      (PERSON id:"stk−mdcl−spct" role:"medical specialist"
            category:@business.user.direct description:"A doctor specialized
            in virology, namely HIV/AIDS and Hepatitis."))
    # a representative of the patients' community
    (PERSON id:"stk−ptnt" role:"patient" name:"Fred Chancey"
          category:@business.user.direct description:"A patient infected with
          HIV/AIDS and/or Hepatitis viruses.")
    # the contractor company and its members
    (ORGANIZATION id:"stk−cntr−cmpn" role:"contractor company"
          name:"SIQuant" category:@technical description:"The company
          that is going to develop and maintain the AIDSPortugal web
          application."
      (PERSON id:"stk−it−admr" role:"IT administrator" name:"Eng. Daniel
            Clawson" description:"An IT professional that configures and
            maintains the AIDSPortugal web application."))))
```

Goals. This viewpoint documents the `Stakeholders`' *intentions*, thus allowing to answer two fundamental questions: (1) "why is this software system needed?", and (2) "how can coarse-grained, high-level business objectives be decomposed into concrete, realizable requirements?". The constructs provided by this viewpoint support the specification of the relations between business objectives (as shown in Listing 5), and how they are derived into requirements that satisfy them. Also, it allows to establish a bridge from the system-of-interest's capabilities to the business context.

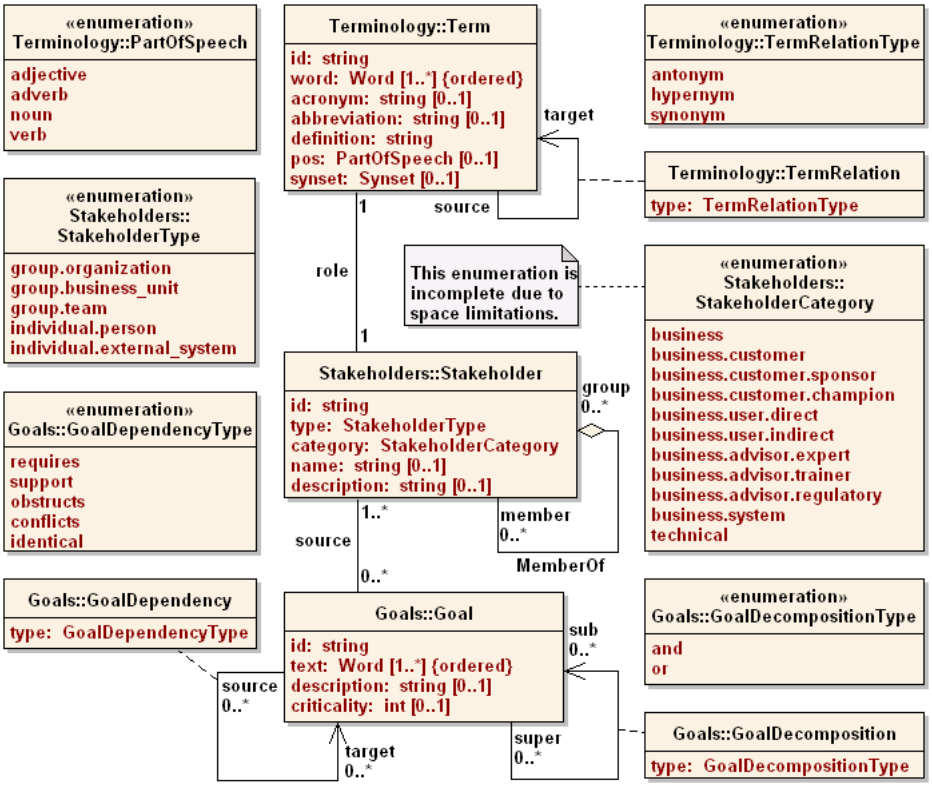

Figure 3. Metamodel of RSL-IL Terminology, Stakeholders, and Goals viewpoints.

Listing 5. Excerpt from the Goals viewpoint of the case study.

```
(PROJECT (...)
  (GOALS
    (GOAL id:"gol-info" text:"To disseminate information about HIV/AIDS and
          Hepatitis." source:"client organization" criticality:@high
          description:"The AIDSPortugal web application must act as a
          web-portal about HIV/AIDS and Hepatitis."
    (DECOMPOSITION type:@and (...)))
    (GOAL id:"gol-advc" text:"To give advice to patients infected with the
          HIV/AIDS and/or Hepatitis viruses." source:"client organization"
          criticality:@high description:"The AIDSPortugal web application
          must provide communication mechanisms so that patients can expose
          their problems."
      (DECOMPOSITION type:@and
        (GOAL id:"gol-advc" text:"To realize the online doctor office metaphor."
              source:"chairman" criticality:@high description:"The
              mechanism through which patients communicate with medical
              specialists must resemble the interactions with doctors in the
              physical reality."
          (DECOMPOSITION type:@or
            (GOAL id:"gol-advc-qa" text:"To give advice to patients through
                  an asynchronous question answering mechanism."
                  source:"chairman" criticality:@high)
            (GOAL id:"gol-advc-im" text:"To give advice to patients through
                  an instant messaging mechanism." source:"chairman"
                  criticality:@low)))))
    (DEPENDENCY type:@supports source:"gol-advc" target:"gol-info")))
```

D. The RSL-IL System Level

After specifying the business-related information, the development team becomes sufficiently acquainted with the problem domain to devise more detailed requirements. These constructs are provided by the RSL-IL System Level viewpoints, which support the specification of both *static* and *dynamic* concerns regarding the system-of-interest, namely: (1) the logical decomposition of a complex system into several system elements, each with their own capabilities and characteristics, thus providing a suitable approach to organize and allocate requirements; (2) the requirements that express the desired features of the system-of-interest, and also the constraints and quality attributes; (3) the data structures aligned with the business jargon, their relations, and a logical description of their attributes; and (4) the actors, functions, event-based state transitions, and use cases that further detail the aforementioned requirements. Considering these concerns, the System Level comprises the following viewpoints: (1) `Architectural`; (2) `Requirements`; (3) `Structural`; and (4) `Behavioral`, respectively.

Although this level also encompasses the `Business Rules` viewpoint, due to space limitations we are unable to include it in this paper. For the same reason, the `Functions` and the `States` parts of the `Behavioral` viewpoint are left out. However, we exemplify the specification of `Actors` and `Use Case` descriptions, since they are rich enough to demonstrate the concrete syntax of RSL-IL.

Architectural. As it can be observed in Figure 4, the purpose of this viewpoint is to provide a *structuring mechanism* to decompose the system-of-interest into its parts, which correspond to the traditional notions of architectural elements such as system, sub-system, and component.

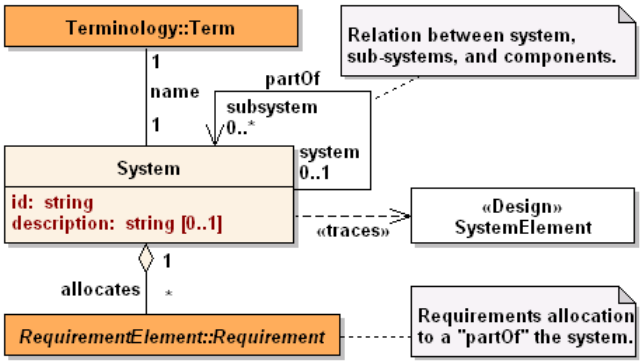

Figure 4. Metamodel of the RSL-IL Architectural viewpoint.

Listing 6 shows the specification of the Online Experts module's scope as a (sub)`System` within the outer `System` that represents the AIDSPortugal web application.

Listing 6. Excerpt from the Architectural viewpoint of the case study.

```
(PROJECT (...)
  (GLOSSARY (...))
  (STAKEHOLDERS (...))
  (GOALS (...))
  (SYSTEM id:"sys-adsp" name:"AIDSPortugal" description:"The top-level
      scope of AIDSPortugal."
    (REQUIREMENTS (...))
    (DOMAIN (...))
    (BEHAVIOR (...))
    (SYSTEM id:"sys-onln-expt" name:"Online Experts" description:"The
        inner scope of Online Experts."
      (REQUIREMENTS (...))
      (DOMAIN (...))
      (BEHAVIOR (...)))))
```

Requirements. This viewpoint focus on the specification of *individual* requirements, and aims to provide a kernel that can be used to convey information about requirements based on the traditional requirements documentation approach. Thus, these constructs are intrinsically related with the support given to natural language "shall" statements, as illustrated in the excerpt presented in Listing 7.

Listing 7. Adornment of a natural language requirement with attributes.

```
{ proposer:"Dr. Harrison Schaffner"; created:"2012-08-16"; priority:"high";
status:"approved" }
FR-3.1.2.4.10 An 'anonymous user' shall be able to ask a new question to a
'medical specialist'.
```

As illustrated in Figure 5, the constructs included in this viewpoint provide the means to: (1) classify individual requirements according to their type (i.e., functional, quality, or constraint); (2) specify their attributes (either intrinsic to them, or required for management purposes); (3) traceability relations; and (4) the rationale history.

Also, this viewpoint includes constructs for explicitly defining the requirements' *semantics* (as depicted in Figure 6), namely to identify the role of each *word* – defined within

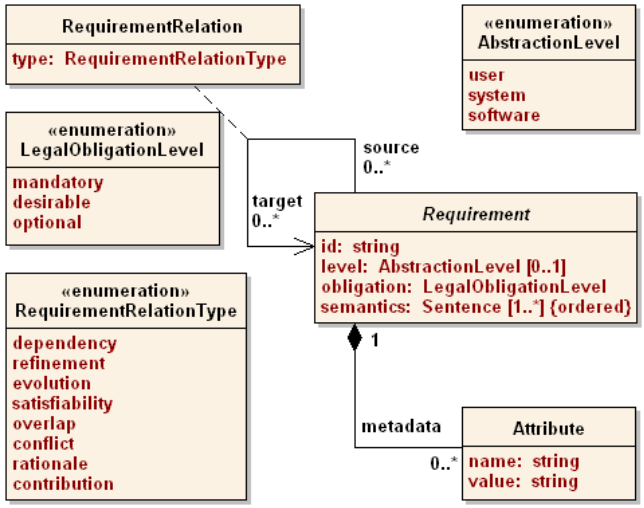

Figure 5. The Requirements/Element part of the Requirements metamodel.

the `Terminology` viewpoint – with regard to the main *head verb* that expresses the system capability referred by the requirement sentence. This information can be automatically extracted from VerbNet[2]. These constructs are the cornerstone to support RSLingo's goal of improving requirements specification quality, through a deeper analysis of the meaning of individual requirement represented as natural language sentences. As it can be observed in Figure 2, the `Requirements` viewpoint provides the "glue" required to connect the `Business Level` with the `System Level` viewpoints of RSL-IL through their mutual dependency upon the `Requirements` viewpoint, which thus acts as a "middle-tier" layer. Thus, user `Requirements` can be regarded as the bridge between the business-oriented `Goals` and the detailed software-oriented `Requirements`, which are expressed in a more technical language. However, and despite the different classifications of requirements according to their abstraction level, to better organize requirements – and avoid ambiguity –, in RSL-IL `Requirements` belong to the RSL-IL System Level and are allocated to `Systems`, as illustrated in Listing 8.

Listing 8. Excerpt from the Requirements viewpoint of the case study.

```
(PROJECT (...)
  (GLOSSARY (...))
  (STAKEHOLDERS (...))
  (GOALS (...))
  (SYSTEM id:"sys-adsp" name:"AIDSPortugal" (...)
    (SYSTEM id:"sys-onln-expt" name:"Online Experts" (...)
      (REQUIREMENTS (...)
        (FUNCTIONAL id:"req-fr-3.1.2.4.5" obligation:@mandatory
          (ATTRIBUTES
            (ATTRIBUTE name:"proposer" value:"Dr. Harrison Schaffner")
            (ATTRIBUTE name:"created" value:"2012-08-16")
            (ATTRIBUTE name:"priority" value:"high")
            (ATTRIBUTE name:"status" value:"approved"))
          (SEMANTICS
            (SENTENCE order:"1" text:"An 'anonymous user' shall be able to
                ask a question to a 'medical specialist'."
```

[2]http://verbs.colorado.edu/~mpalmer/projects/verbnet.html (accessed on January 25th, 2013)

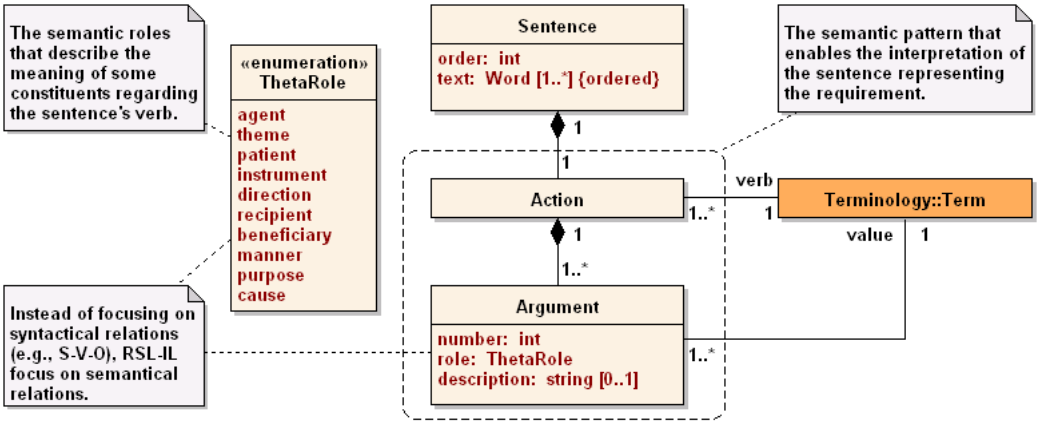

Figure 6. Metamodel of the Requirements/Semantics part of the RSL-IL Requirements viewpoint.

```
(ACTION verb:"ask" # extracted from PropBank corpus, aligned
      with VerbNet
   (ARG n:"0" role:@agent description:"asker" value:"anonymous
      user")
   (ARG n:"1" role:@topic description:"question" value:"question")
   (ARG n:"2" role:@recipient description:"hearer" value:"medical
      specialist")))))))))
```

Listing 8 clearly illustrates the kind of consistency checking supported by RSL-IL. By identifying the *noun phrases* that map to specific semantic roles acting as parameters of the main verb of the requirement sentence, one is able to indirectly determine through the Terminology viewpoint whether these noun phrases are semantically suitable to fill these roles.

Structural. Since the system-of-interest must store and manipulate information about entities of the problem space (i.e., the "as-is"), this viewpoint deals with the *data perspective* of requirements by providing the means to describe the concepts (and their relations) pertaining to the *domain model* of the system, as illustrated in Figure 7.

Listing 9. Excerpt of entities definition in natural language.

A question entity consists of:
– A 'question text' (a text field);
– A 'creation date' (a datetime field);
– An 'author name' (a text field);
– An 'author e-mail' (a regex field);
– A 'medical specialist' (refers to a 'medical specialist' entity);
– A 'question topic' (refers to a 'question topic' entity);
– A 'hits' (a integer field);
– A 'publishing state' (an enumeration field);
– An 'object state' (an enumeration field).

According to the 'publishing state' attribute, a question entity can be in one of the following values: 'pending approval', 'approved and private', 'approved and public', or 'approved, public, and FAQ'.

According to the 'object state' attribute, a question entity can be in one of the following values: 'active', or 'inactive'.

Listing 10 illustrates how these constructs can be combined together to enable the detailed documentation of structural concerns, namely the specification of the entity described in the natural language excerpt presented in Listing 9.

Listing 10. Excerpt from the Structural viewpoint of the case study.

```
(PROJECT (...)
  (SYSTEM id:"sys-adsp" name:"AIDSPortugal" (...)
    (SYSTEM id:"sys-onln-expt" name:"Online Experts" (...)
      (DOMAIN
        (ENTITY id:"ent-qstn" name:"question" description:"A question posed
            by an anonymous user."
          (ATTRIBUTE name:"question text" type:@text)
          (ATTRIBUTE name:"creation date" default:"!datetime.now"
              type:@datetime description:"The date and time at which the
              question was made.")
          (ATTRIBUTE name:"author name" type:@text)
          (ATTRIBUTE name:"author e-mail" type:@regex
              format:"[a-z0-9]+(\.[a-z0-9]+)*@([a-z]+\.)+[a-z]{2,4}"
              description:"It must be in lowercase and does not allow
              underscores in the local-part.")
          (ATTRIBUTE name:"medical specialist" type:@ref entity:"medical
              specialist" lower:"0" upper:"1")
          (ATTRIBUTE name:"question topic" type:@ref entity:"question topic"
              lower:"0" upper:"1")
          (ATTRIBUTE name:"hits" type:@integer default:"0" description:"The
              counter with the number of times a question was viewed.")
          (ATTRIBUTE name:"publishing state" type:@enumeration value:"{
              pending_approval, approved_private, approved_public,
              approved_public_faq }" default:"pending_approval"
              description:"A question must be first approved. Afterward, it
              can be classified either private or public. Finally, a public
              question can be further classified as being a FAQ.")
          (ATTRIBUTE name:"object state" type:@enumeration value:"{ active,
              inactive }" default:"active" description:"By default a question is
              active. However, it can become inactive to appear as it was
              deleted, but without losing historical information.")
          (ATTRIBUTE name:"answers" type:@ref entity:"answer" lower:"0"
              upper:"*"))
        (...))))) # the business entities 'question topic', 'answer', and 'medical
            specialist' are omitted
```

Behavioral/Actors. As depicted in Figure 8, this part of the Behavior viewpoint handles the representation of `Actors`, the System Level counterparts of `Stakeholders`, which are related with them through a common `Term`. An `Actor` represents the role played by a `Stakeholder` while interacting with the system. Listing 11 exemplifies how these constructs can be applied to specify some of the external entities that interact with the Online Experts module.

Behavioral/UseCases. This viewpoint is aligned with one of the most versatile techniques to specify complex functional requirements. Despite its simplicity, `UseCases` provides a

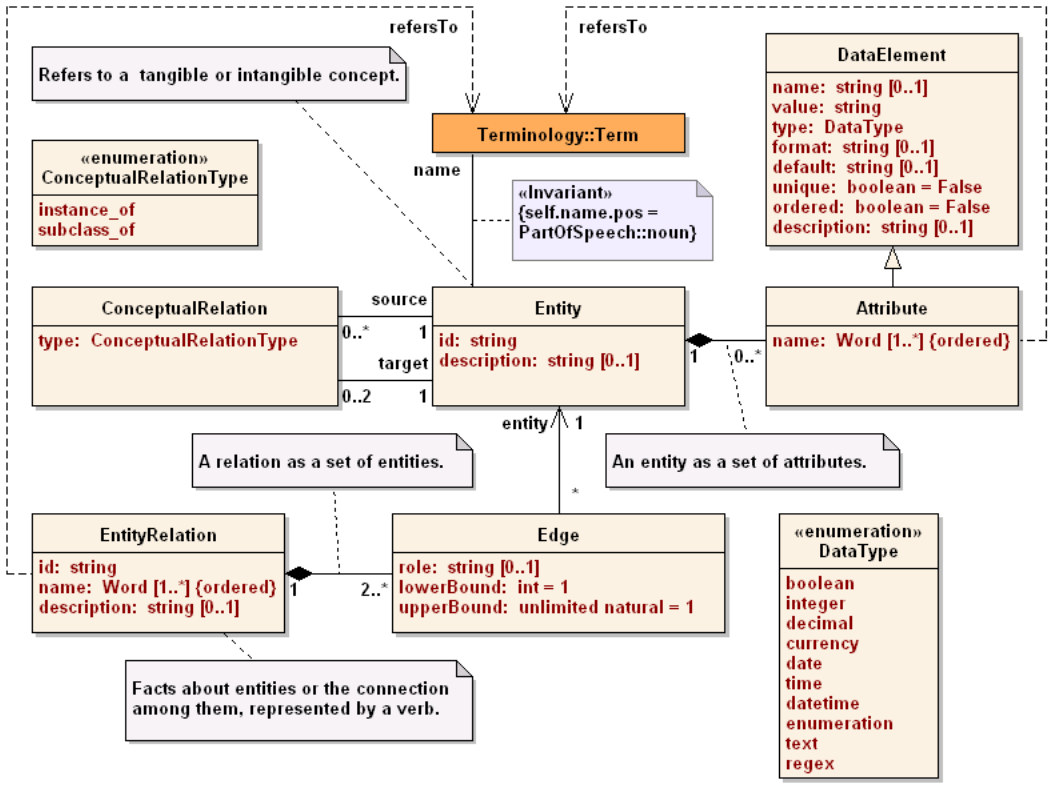

Figure 7. Metamodel of the RSL-IL Structural viewpoint.

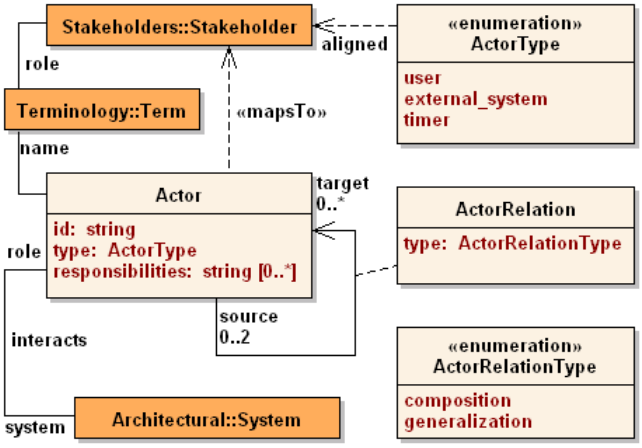

Figure 8. The Behavioral/Actors part of the Behavioral metamodel.

practical way to describe and analyze (part of) the behavior of the system-of-interest based on the concept of *scenario*. Through the linguist analysis of all `UseCases` (i.e., descriptions of behavior fragments) one is able to gather a thorough description of the external behavior of the system-of-interest, according to the different perspectives of business stakeholders. Listing 12 exemplifies the specification of a `UseCase` in RSL-IL, as well as its interdependencies on several RSL-IL constructs of other viewpoints. It is noteworthy that, since each `UseCaseStep` is represented by a natural language sentence, the role-based semantic analysis applied to `Requirements` can also be performed.

Listing 11. Excerpt from the Behavioral/Actors viewpoint of the case study.

```
(PROJECT (...)
  (SYSTEM id:"sys-adsp" name:"AIDSPortugal" (...)
    (BEHAVIOR
      (ACTORS (...) # common actors are factored out to avoid redundancy
        (ACTOR id:"act-anms-user" name:"anonymous user"
               responsibilities:"Navigate through the system to consult public
               information. Interact with other users through the system."))
    (SYSTEM id:"sys-onln-expt" name:"Online Experts" (...)
      (BEHAVIOR
        (ACTORS
          (ACTOR id:"act-mdcl-spct" name:"medical specialist"
            (ACTOR-SPECIALIZATIONS
              (ACTOR id:"act-mdrr" name:"moderator")))
          (ACTOR-RELATION type:@generalization source:"act-rgu"
                target:"act-mdcl-spct")))))))
```

Listing 12. Excerpt of case study's Behavioral/UseCases viewpoint.

```
(PROJECT (...)
  (SYSTEM id:"sys-adsp" name:"AIDSPortugal" (...)
    (SYSTEM id:"sys-onln-expt" name:"Online Experts" (...)
      (BEHAVIOR
        (USE-CASES
          (USE-CASE id:"uc-ask-qstn" name:"ask a question"
                description:"The interaction through which an anonymous user
                is able to pose a question to a medical specialist."
            (ACCOMPLISHES goal:"gol-advc-qa")
            (RELATED-WITH functional:"FR-3.1.2.4.10")
```

```
    (ACTORS initiates:"act−anms−user" participates:"act−mdrr")
    (TRIGGERED−BY event:"evt−ask−qstn")
    (CONDITIONS
      (PRE (CONDITION usecase:"uc−nvgt−onln−exprt"))
      (POST
        (CONDITION entity:"ent−qstn" attribute:"publishing state"
            current:"pending_approval")
        (CONDITION entity:"ent−qstn" attribute:"object state"
            current:"active")))
    (REQUIRES entity:"ent−qstn")
  (SCENARIOS
    (SCENARIO name:"Default ask question procedure." type:@normal
        sequential:"true"
      (STEP label:"1" text:"The 'anonymous user' selects the 'ask
          question' functionality.")
      (STEP label:"2" text:"The system asks the 'anonymous user' to
          insert: the 'question text' (mandatory), the 'author name'
          (mandatory), and the 'author e−mail' (mandatory).")
      (STEP label:"3" text:"The 'anonymous user' fills the question
          form's fields.")
      (STEP label:"4" text:"The system asks the 'anonymous user' to
          select the 'medical specialist' to whom the question should
          be directed.")
      (STEP label:"5" text:"The 'anonymous user' selects one of the
          available 'medical specialists'.")
      (STEP label:"6" text:"The system challenges the 'anonymous
          user' to provide evidence that he/she is a human being.")
      (STEP label:"7" text:"The 'anonymous user' meets the challenge.")
      (STEP label:"8" text:"The system validates the new question's
          fields.")
      (STEP label:"9" text:"The system notifies the moderator.")
      (STEP label:"10" text:"The system displays a success message to
          the 'anonymous user'."))
    (SCENARIO name:"Exceptions" type:@exception sequential:"true"
      (STEP label:"7.1a" text:"The 'anonymous user' fails the
          challenge.")
      (CONTINUE label:"7.1b" next:"6" description:"The system
          continues until the 'anonymous user' meets the
          challenge.")))))))))
```

Although we did not introduce all System Level constructs, those that were are sufficient to overview RSL-IL, and demonstrate its benefits in terms of requirements *consistency checking* and *ambiguity resolution*, which are achieved through a tight integration between the different RSL-IL viewpoints by means of well-defined `Terms` with rich semantic definitions.

IV. Discussion

Most RSLs available today can be broadly classified as being either: unconstrained natural languages, controlled natural languages, graphical modeling languages, or formal method languages. *Unconstrained natural languages* is the most common requirement specification approach, namely for software-intensive systems that are not mission critical. Since this RSL cannot yet be fully handled by computers due to the current limitations of NLP techniques, the quality assurance of requirements specifications documented with unconstrained natural language consists mainly in guidelines and template-based formats that synthesize the best practices in terms of requirements organization (e.g., document outline) and writing style (e.g., linguistic patterns) [1]. To resolve the ambiguity, and also the computability, problems one can resort to *controlled natural languages* (e.g., Attempto Controlled English [11]), which are subsets of natural language with a limited vocabulary and a well-defined set of syntactic patterns. However, since these engineered languages have a strict syntax – although they appear to be natural language – they are hard to author without specialized tools (e.g., predictive text editors [12], [13]). On the other hand we have *graphical modeling languages* (e.g., UML, SysML, ReqIF) that support the documentation of requirements as conceptual models. However, these languages are regarded as semi-formal and their current tool support do not strictly enforce their semantics. Thus, people tend to use them as sketching tools do create a set of scattered diagrams, instead of modeling the requirements of the system-of-interest as a coherent model composed of a set of integrated diagrams. Finally, *formal method languages* (such as Z, VDM, the B-Method) are based on mathematical and logical foundations (e.g., set theory, first-order logic), and provide a very rigorous and formal way to specify (parts of) systems [14]. However, this requirements formalization approach is expensive to implement because even stakeholders with a technical background might require training on the mathematical/logical foundations of these languages. Thus, the models created with these languages are closer to software design concerns, which allows – in some cases – the stepwise refinement of the original specifications to generate source code. Consequently, given that they are hard to be understood by business stakeholders, these languages tend to be used only for mission-critical systems, such as those required by the military, health, or transport industries.

Considering these main categories of RSLs, we classify RSL-IL as a *formal language* because – as programming languages – it consists of a set of constructs with well-defined semantics, and is computer-processable through a notation that can be represented through a context-free grammar. However, we do not claim that RSL-IL is a formal method language. The design principles of RSL-IL are distinct from those of formal method languages, which are more oriented toward the mathematical/logical representation, thus closer of software design (at the solution space) rather than focusing on requirements (at the problem space). For this reason, purists of formal languages might argue that RSL-IL is a semi-formal modeling language with textual notation, because RSL-IL constructs follow an object-oriented representation approach, which they consider as having "less formal" (neither mathematical, nor logical) underpinnings. Nevertheless, RSL-IL was designed to be formal and computer-processable, namely to enable the automation of requirements verification (i.e., sanity checks) of their quality criteria [5], but without loosing its simplicity nor its focus on RE concerns. For instance, since RSL-IL deals with individual requirements at a deeper semantic level to address their meaning (instead of only handling their metadata), RSL-IL constructs foster – by construction – ambiguity resolution, consistency checking, and completeness enforcement. The results of this discussion, namely the assessment of RSL-IL in relation to other RSLs, are summarized in Table I.

The effort to develop RSL-IL is justified by the following rationale. First, the unique characteristics of the RSLingo approach required a specific language for formally representing requirements information, extracted through the alignment of linguistic patterns encoded in RSL-PL [6]. Second, RSL-IL provides an intermediate representation of requirements that, by raising the level of formality, becomes amenable to be

Table I
COMPARISON OF RSL-IL WITH OTHER RSLS.

RSL Design Principles	**UNL**	**ACE**	**UML**	**B**	**RSL-IL**
Theoretical Foundations					
Purpose	Any	Knowledge	Software	Software	RE
Formality	Low	High	Medium	Highest	High
Abstraction Levels	Both	Business	Both	System	Both
Computer-processable	Partial	Yes	Yes	Yes	Yes
Quality Enforcement					
Clearness	Manual	Automatic	Manual	Automatic	Automatic
Consistency	Manual	Manual	Manual	Automatic	Automatic
Completeness	Manual	Manual	Manual	Manual	Automatic
Usability					
Learnability	Easy	Medium	Medium	Hard	Medium
Readability	Easy	Easy	Medium	Hard	Medium
Writability	Easy	Hard	Medium	Hard	Medium

employed in automatic requirements verification tasks, and for enabling the generation of complementary views based on the information it conveys. Third, RSL-IL can also be used by a generative process as a source language within the context of model-to-model transformations. Such transformations would allow one to represent the same requirements baseline (encoded in RSL-IL) into another target language, such as a UML profile or a controlled natural language [11].

V. CONCLUSION

RSL-IL is a requirements-oriented specification language designed to support the back-end mechanisms of the RSLingo approach, namely to automate some requirements verification tasks. Thus, the information extraction techniques employed by RSLingo warrants a formal language, such as RSL-IL, for conveying in a computer-processable manner the information extracted from natural language requirements.

This paper presents most of RSL-IL constructs, as well as the schema according to which they are internally organized into specific viewpoints that – when combined – enable requirements engineers to specify both Business and System Level concerns of the system-of-interest. Also, this paper provides a running example that, besides better explaining the concepts underlying the RSL-IL constructs encompassed by these viewpoints, illustrates how they can be applied in practice. Finally, we compare RSL-IL with other RSLs to discuss its advantages based on the insight provided by the running example.

As future work, we plan two different research paths. On one hand, we intent to assess usability concerns of RSL-IL to determine how easy it is to read and write RSL-IL specifications, both in terms of cognitive alignment with the notions used by common RE techniques, and also in terms of the usability of its concrete syntax. On the other hand, we plan to research further heuristics and rules to automatically increase the quality of requirements specifications.

ACKNOWLEDGMENT

This work was supported by national funds through FCT – Fundação para a Ciência e a Tecnologia, under the PEst-OE/EEI/LA0021/2013 project.

REFERENCES

[1] K. Pohl, *Requirements Engineering: Fundamentals, Principles, and Techniques*, 1st ed. Springer, July 2010, ISBN-13: 978-3642125775.
[2] K. Emam and A. Koru, "A Replicated Survey of IT Software Project Failures," *IEEE Software*, vol. 25, no. 5, pp. 84–90, September 2008.
[3] A. Davis, *Just Enough Requirements Management: Where Software Development Meets Marketing*, 1st ed. Dorset House Publishing, May 2005.
[4] B. Kovitz, *Practical Software Requirements: Manual of Content and Style*. Manning, July 1998.
[5] IEEE Computer Society, "IEEE Recommended Practice for Software Requirements Specifications," *IEEE Std 830-1998*, August 1998.
[6] D. Ferreira and A. Silva, "RSLingo: An Information Extraction Approach toward Formal Requirements Specifications," in *Proc. of the 2nd Int. Workshop on Model-Driven Requirements Engineering (MoDRE 2012)*. IEEE Computer Society, September 2012, pp. 39–48, ISBN-13: 978-1-4673-4388-6.
[7] H. Cunningham, "Information Extraction, Automatic," in *Encyclopedia of Language & Linguistics*, 2nd ed. Elsevier, 2006, vol. 5, pp. 665–677.
[8] D. Ferreira and A. Silva, "RSL-PL: A Linguistic Pattern Language for Documenting Software Requirements," in *Proc. of the 3rd International Workshop on Requirements Patterns (RePa 2013)*. IEEE Computer Society, July 2013.
[9] T. Smith and M. Waterman, "Identification of common molecular subsequences," *Journal of Molecular Biology*, vol. 147, no. 1, pp. 195–197, March 1981.
[10] G. Miller, "WordNet: A Lexical Database for English," *Communications of the ACM*, vol. 38, no. 3, pp. 39–41, 1995.
[11] N. Fuchs, U. Schwertel, and R. Schwitter, "Attempto Controlled English – Not Just Another Logic Specification Language," in *Logic-Based Program Synthesis and Transformation*, P. Flener, Ed., Eighth International Workshop LOPSTR'98. Manchester, UK: Springer, June 1999, pp. 1–20.
[12] T. Kuhn, "AceWiki: A Natural and Expressive Semantic Wiki," in *Semantic Web User Interaction at CHI 2008: Exploring HCI Challenges*. CEUR Workshop Proceedings, 2008.
[13] ——, "AceWiki: Collaborative Ontology Management in Controlled Natural Language," in *3rd Semantic Wiki Workshop*. CEUR Workshop Proceedings, 2008.
[14] I. Sommerville and P. Sawyer, *Requirements Engineering: A Good Practice Guide*. Wiley, April 1997.